Events

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

Function Pointer Examples

```
void main(void) {

   int x = 10;
   int y = 20;
   int r1, r2, r3, r4;

   int (*func) (int, int);

   func = &add;
   r1 =  func(x, y);

   func = &sub;
   r2 =  func(x, y);

   func = &mul;
   r3 =  func(x, y);

   func = &div;
   r4 =  func(x, y);

   printf("x= %d, y= %d\n", x, y);
   printf("r1= %d\n", r1);
   printf("r2= %d\n", r2);
   printf("r3= %d\n", r3);
   printf("r4= %d\n", r4);


   printf("add= %p &add= %p\n", add, &add);
   printf("sub= %p &sub= %p\n", sub, &sub);
   printf("mul= %p &mul= %p\n", mul, &mul);
   printf("div= %p &div= %p\n", div, &div);


}
```

```
#include <stdio.h>

int add(int a, int b) { return (a + b); }
int sub(int a, int b) { return (a - b); }
int mul(int a, int b) { return (a * b); }
int div(int a, int b) { return (a / b); }
```

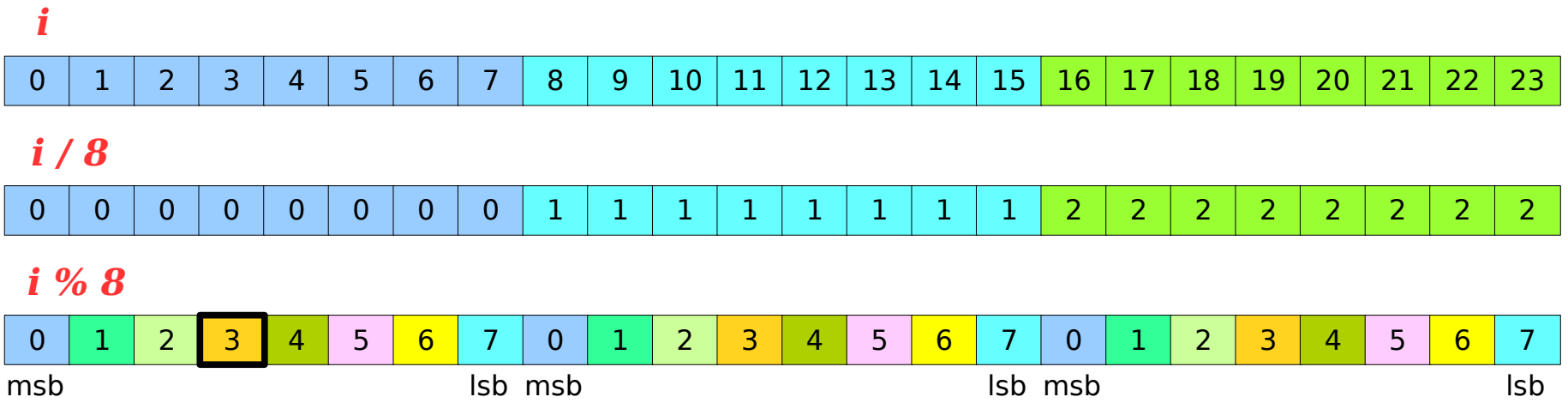
```
young@USB01:~$ ./a.out
x= 10, y= 20
r1= 30
r2= -10
r3= 200
r4= 0
add= 0x40055d &add= 0x40055d
sub= 0x400571 &sub= 0x400571
mul= 0x400587 &mul= 0x400587
div= 0x40059a &div= 0x40059a
```

Array Name and Function Name

```c
#include <stdio.h>

int func(int x) { return (x+2); }

int main(void) {
  int i = 2;
  int a[2] = {22, 33};

  printf("&i=%p, i=%d \n\n", &i, i);
  printf("&a=%p, a=%p \n\n", &a, a);
  printf("a=%p, *a=%d \n", a, *a);
  printf("a+1=%p, *(a+1)=%d \n\n", a+1, *(a+1));

  printf("&func=%p, func=%p \n\n", &func, func);
  printf("&main=%p, main=%p \n\n", &main, main);

}
```

```
young@USB01:~$ ./a.out
&i=0x7ffd49b10f9c, i=2

&a=0x7ffd49b10fa0, a=0x7ffd49b10fa0

a=0x7ffd49b10fa0, *a=22
a+1=0x7ffd49b10fa4, *(a+1)=33

&func=0x40052d, func=0x40052d

&main=0x40053c, main=0x40053c
```

Manual Packing



0x80

1	0	0	0	0	0	0	0

0x80 >> (i % 8)

0	0	0	1	0	0	0	0

Assume ***i % 8 = 3***

X

x	x	x	x	x	x	x	x

0x80 >> (i % 8)

0	0	0	1	0	0	0	0

X |= (0x80 >> (i % 8))

x	x	x	1	x	x	x	x

Set

X

x	x	x	x	x	x	x	x

~(0x80 >> (i % 8))

1	1	1	0	1	1	1	1

X &= ~(0x80 >> (i % 8))

x	x	x	0	x	x	x	x

Clr

Trigger.h and Trigger.c

Based on

Implementation of generic triggers.
Erich Styger, erich.styger@hslu.ch
21.03.2011

http://www.steinerberg.com/EmbeddedComponents/BookSrc/Event.c
http://www.steinerberg.com/EmbeddedComponents/BookSrc/Event.h

Overview of Trigger.h

```
typedef enum EVNT_Handle { ??? } EVNT_Handle;

void EVNT_SetEvent(EVNT_Handle event);

void EVNT_ClearEvent(EVNT_Handle event);

bool EVNT_EventIsSet(EVNT_Handle event);

void EVNT_HandleEvent( void (*callback) (EVNT_Handle) );

void EVNT_Init(void);

#en
```

Overview of Trigger.c

```
typedef uint8_t EVNT_MemUnit;
#define EVNT_MEM_UNIT_NOF_BITS

static EVNT_MemUnit EVNT_Events[ ??? ];

#define SET_EVENT(event)
#define CLR_EVENT(event)
#define GET_EVENT(event)

void EVNT_SetEvent(EVNT_Handle event) { ??? }
void EVNT_ClearEvent(EVNT_Handle event) { ??? }
bool EVNT_EventIsSet(EVNT_Handle event) { ??? }
void EVNT_HandleEvent(void (*callback)(EVNT_Handle)) { ??? }
void EVNT_Init(void) { ??? }
```

Event.h

```
typedef enum EVNT_Handle {
  EVNT_INIT,
  EVNT_SW1_PRESSED,
  EVNT_SW2_PRESSED,
  EVNT_SW3_PRESSED,
  EVNT_SW4_PRESSED,
  EVNT_NOF_EVENTS
} EVNT_Handle;

void EVNT_SetEvent(EVNT_Handle event);

void EVNT_ClearEvent(EVNT_Handle event);

bool EVNT_EventIsSet(EVNT_Handle event);

void EVNT_HandleEvent(void (*callback)(EVNT_Handle));

void EVNT_Init(void);
```

Event.c – Macros

```
typedef uint8_t EVNT_MemUnit;
#define EVNT_MEM_UNIT_NOF_BITS  (sizeof(EVNT_MemUnit)*8)

static EVNT_MemUnit EVNT_Events[      \
         ((EVNT_NOF_EVENTS-1)/EVNT_MEM_UNIT_NOF_BITS)+1   \
];

#define SET_EVENT(event) \
  EVNT_Events[(event)/EVNT_MEM_UNIT_NOF_BITS] |=      \
       (1<<(EVNT_MEM_UNIT_NOF_BITS-1))        >>      \
       ((uint8_t)((event)%EVNT_MEM_UNIT_NOF_BITS))

#define CLR_EVENT(event) \
  EVNT_Events[(event)/EVNT_MEM_UNIT_NOF_BITS] &=      \
      ~((1<<(EVNT_MEM_UNIT_NOF_BITS-1))       >>      \
       ((uint8_t)((event)%EVNT_MEM_UNIT_NOF_BITS)))

#define GET_EVENT(event) \
  (bool)(EVNT_Events[(event)/EVNT_MEM_UNIT_NOF_BITS] &  \
         ((1<<(EVNT_MEM_UNIT_NOF_BITS-1))            >> \
         ((uint8_t)((event)%EVNT_MEM_UNIT_NOF_BITS))))
```

Event.c – Set(), Clear(), IsSet()

```
void EVNT_SetEvent(EVNT_Handle event) {
  EnterCritical();
  SET_EVENT(event);
  ExitCritical();
}

void EVNT_ClearEvent(EVNT_Handle event) {
  EnterCritical();
  CLR_EVENT(event);
  ExitCritical();
}

bool EVNT_EventIsSet(EVNT_Handle event) {
  bool isSet;

  EnterCritical();
  isSet = GET_EVENT(event);
  ExitCritical();
  return isSet;
}
```

Event.c – HandleEvent( )

```
void EVNT_HandleEvent(void (*callback)(EVNT_Handle)) {
   EVNT_Handle event;

   EnterCritical();
   for (event=(EVNT_Handle)0; event<EVNT_NOF_EVENTS; event++) {

     if (GET_EVENT(event)) {
       CLR_EVENT(event);
       break;
     }
   }
   ExitCritical();
   if (event != EVNT_NOF_EVENTS) {
     callback(event);
   }
}
```

Trigger.c – Init()

```
void EVNT_Init(void) {
  uint8_t i;

  i = 0;
  do {
    EVNT_Events[i] = 0;
    i++;
  } while( i <  sizeof(EVNT_Events) /          \
                sizeof(EVNT_Events[0])         \
         );
  EVNT_SetEvent(EVNT_INIT);
}
```

References

[1] http://wiki.osdev.org/ARM_RaspberryPi_Tutorial_C
[2] http://blog.bobuhiro11.net/2014/01-13-baremetal.html
[3] http://www.valvers.com/open-software/raspberry-pi/
[4] https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/os/downloads.html